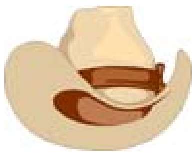
Latest Release: 1.5.2

Download
Installation
Basic Usage
Advanced Usage
Credits & Copyright
Change Log
Project Page
Mailing Lists
Bug Tracker

Last Update: 12-31-09



The OpenGL Extension Wrangler Library

Initializing GLEW

First you need to create a valid OpenGL rendering context and call `glewInit()` to initialize the extension entry points. If `glewInit()` returns `GLEW_OK`, the initialization succeeded and you can use the available extensions as well as core OpenGL functionality. For example:

```
#include <GL/glew.h>
#include <GL/glut.h>
...
glutInit(&argc, argv);
glutCreateWindow("GLEW Test");
GLenum err = glewInit();
if (GLEW_OK != err)
{
  /* Problem: glewInit failed, something is seriously
wrong. */
  fprintf(stderr, "Error: %s\n",
glewGetErrorString(err));
  ...
}
fprintf(stdout, "Status: Using GLEW %s\n",
glewGetString(GLEW_VERSION));
```

Checking for Extensions

Starting from GLEW 1.1.0, you can find out if a particular extension is available on your platform by querying globally defined variables of the form `GLEW_{extension_name}`:

```
if (GLEW_ARB_vertex_program)
{
  /* It is safe to use the ARB_vertex_program extension
here. */
  glGenProgramsARB(...);
}
```

In GLEW 1.0.x, a global structure was used for this task. To ensure binary compatibility between releases, the struct was replaced with a set of variables.

You can also check for core OpenGL functionality. For example, to see if OpenGL 1.3 is supported, do the following:

```
if (GLEW_VERSION_1_3)
{
  /* Yay! OpenGL 1.3 is supported! */
}
```

In general, you can check if `GLEW_{extension_name}` or `GLEW_VERSION_{version}` is true or false.

It is also possible to perform extension checks from string input. Starting from the 1.3.0 release, use `glewIsSupported` to check if the required core or extension functionality is available:

```
if
(glewIsSupported("GL_VERSION_1_4  GL_ARB_point_sprite"))
{
  /* Great, we have OpenGL 1.4 + point sprites. */
}
```

For extensions only, `glewGetExtension` provides a slower alternative (GLEW 1.0.x-1.2.x). **Note that in the 1.3.0 release** `glewGetExtension` **was**

replaced with glewIsSupported.

```
if (glewGetExtension("GL_ARB_fragment_program"))
{
  /* Looks like ARB_fragment_program is supported. */
}
```

Experimental Drivers

GLEW obtains information on the supported extensions from the graphics driver. Experimental or pre-release drivers, however, might not report every available extension through the standard mechanism, in which case GLEW will report it unsupported. To circumvent this situation, the glewExperimental global switch can be turned on by setting it to GL_TRUE before calling glewInit(), which ensures that all extensions with valid entry points will be exposed.

Platform Specific Extensions

Platform specific extensions are separated into two header files: wglew.h and glxew.h, which define the available WGL and GLX extensions. To determine if a certain extension is supported, query WGLEW_{extension name} or GLXEW_{extension_name}. For example:

```
#include <GL/wglew.h>

if (WGLEW_ARB_pbuffer)
{
  /* OK, we can use pbuffers. */
}
else
{
  /* Sorry, pbuffers will not work on this platform. */
}
```

Alternatively, use wglewIsSupported or glxewIsSupported to check for extensions from a string:

```
if (wglewIsSupported("WGL_ARB_pbuffer"))
{
  /* OK, we can use pbuffers. */
}
```

Utilities

GLEW provides two command-line utilities: one for creating a list of available extensions and visuals; and another for verifying extension entry points.

visualinfo: extensions and visuals

visualinfo is an extended version of glxinfo. The Windows version creates a file called visualinfo.txt, which contains a list of available OpenGL, WGL, and GLU extensions as well as a table of visuals aka. pixel formats. Pbuffer and MRT capable visuals are also included. For additional usage information, type visualinfo -h.

glewinfo: extension verification utility

glewinfo allows you to verify the entry points for the extensions supported on your platform. The Windows version reports the results to a text file called glewinfo.txt. The Unix version prints the results to stdout.

Windows usage:

```
glewinfo [-pf <id>]
```

where <id> is the pixel format id for which the capabilities are displayed.

Unix usage:

```
glewinfo [-display <dpy>] [-visual <id>]
```

where <dpy> is the X11 display and <id> is the visual id for which the capabilities are displayed.